Package 'spFSR'

November 13, 2022

Type Package

Title Feature Selection and Ranking via Simultaneous Perturbation Stochastic Approximation

Version 2.0.3

Date 2022-10-28

Description
An implementation of feature selection, weighting and ranking via simultaneous perturbation stochastic approximation (SPSA). The SPSA-FSR algorithm searches for a locally optimal set of features that yield the best predictive performance using some error measures such as mean squared error (for regression problems) and accuracy rate (for classification problems).

License GPL-3

Encoding UTF-8

Depends mlr3 (>= 0.14.0), future (>= 1.28.0), tictoc (>= 1.0)

Imports mlr3pipelines (>= 0.4.2), mlr3learners (>= 0.5.4), ranger (>= 0.14.1), parallel (>= 3.4.2), ggplot2 (>= 2.2.1), lgr (>= 0.4.4)

Suggests caret (>= 6.0), MASS (>= 7.3)

URL https://www.featureranking.com/

BugReports https://github.com/yongkai17/spFSR/issues

RoxygenNote 7.2.1

NeedsCompilation no

Author David Akman [aut, cre],
Babak Abbasi [aut, ctb],
Yong Kai Wong [aut, ctb],
Guo Feng Anders Yeo [aut, ctb],
Zeren D. Yenice [ctb]

Maintainer David Akman <david.v.akman@gmail.com>

Repository CRAN

Date/Publication 2022-11-13 09:30:02 UTC

R topics documented:

`getBestModel` *Extracting the wrapped model of the best performing features from a spFSR object*

Description

A fitted model uses the best performing feature subsets. It inherits all methods or functions applied to a `WrappedModel` objects. For example, the `predict` function can be used on the fitted model. See spFeatureSelection for example.

Usage

```
getBestModel(x)
```

Arguments

`x`	a `spFSR` object

Value

A `WrappedModel` object of the best performing features.

See Also

spFeatureSelection

getImportance *Extracting feature importance data from a spFSR object*

Description

This returns importance ranks of best performing features. See spFeatureSelection for example.

Usage

```
getImportance(x)
```

Arguments

x	a spFSR object

Value

A data.frame of features and feature importance

See Also

plotImportance and spFeatureSelection.

plot.spFSR *Ploting a spFSR object*

Description

Plot for a spFSR object. It provides a scatterplot of scoring values vs. iteration. The error bar of scoring values at each iteration can be included. It also allows user to identify the iteration which yields the best scoring value. See spFeatureSelection for example.

Usage

```
## S3 method for class 'spFSR'
plot(x, errorBar = FALSE, annotateBest = FALSE, se = FALSE, ...)
```

Arguments

x	a spFSR object
errorBar	If TRUE, an error bar of +/- 1 standard deviation will be included around the mean error scoring value at each iteration. When it is TRUE, the ylim argument cannot be used. The default is FALSE.
annotateBest	If TRUE, the best result will be highlighted and annotated. The default is FALSE.

se	If TRUE, an error bar of ± standard error will be included around the mean error scoring value at each iteration. When it is TRUE, the ylim argument cannot be used. The se does not produce any error bar if errorBar is set as FALSE. Note that if the standard error is used, the error bar has a narrower range. The default is FALSE.
...	Additional plot parameters that can be passed into the plot function.

Value

Plot error scoring values vs iterations of a spFSR object with an error bar (if included).

See Also

plotImportance and spFeatureSelection.

`plotImportance` *Ploting importance ranks of best performing features from a spFSR object*

Description

A vertical bar chart of features vs. feature importance. See spFeatureSelection for example.

Usage

```
plotImportance(x, low = "darkblue", high = "black")
```

Arguments

x	a spFSR object
low	Color for the lowest importance. The default is darkblue.
high	Color for the highest importance. The default is black.

Value

a ggplot object: a vertical bar chart of features and feature importance.

See Also

plotImportance, spFSR.default, and spFeatureSelection.

spFeatureSelection *SPSA-FSR for Feature Selection and Ranking*

Description

This function searches for the best performing features and rank the feature importance by implementing simultaneous perturbation stochastic approximation (SPSA) algorithm given a task and a wrapper. The task and wrapper are defined using the **mlr3** package.

Usage

```
spFeatureSelection(task, wrapper = NULL, scoring = NULL, ...)
```

Arguments

task	A task object created using **mlr3** package. It must be either a ClassifTask or RegrTask object.
wrapper	A Learner object created using **mlr3** package. Multiple learners object is not supported.
scoring	A performance measure within the **mlr3** package supported by the task.
...	Additional arguments. For more details, see spFSR.default.

Value

spFSR returns an object of class "spFSR". An object of class "spFSR" consists of the following:

task.spfs	An **mlr3** package tsk object defined on the best performing features.
wrapper	An **mlr3** package lrn object, default is random forest.
scoring	An **mlr3** package msr object as specified by the user.
param best.model	An **mlr3** package model object trained by the wrapper using task.spfs.
iter.results	A data.frame object containing detailed information on each iteration.
features	Names of the best performing features.
num.features	The number of best performing features.
importance	A vector of importance ranks of the best performing features.
total.iters	The total number of iterations executed.
best.iter	The iteration where the best performing feature subset was encountered.
best.value	The best measure value encountered during execution.
best.std	The standard deviation corresponding to the best measure value encountered.
run.time	Total run time in minutes
results	Dataframe with boolean of selected features, names and measure
call	Call

References

David V. Akman et al. (2022) k-best feature selection and ranking via stochastic approximation, *Expert Systems with Applications*, **Vol. 213**. See doi:10.1016/j.eswa.2022.118864

G.F.A Yeo and V. Aksakalli (2021) A stochastic approximation approach to simultaneous feature weighting and selection for nearest neighbour learners, *Expert Systems with Applications*, **Vol. 185**. See doi:10.1016/j.eswa.2021.115671

See Also

tsk, lrn, msr and spFSR.default.

Examples

```
library(mlr3)          # load the mlr3 package
library(mlr3learners) # load the mlr3learners package

task    <- tsk('iris') # define task
wrapper <- lrn('classif.rpart')               # define wrapper
measure <- msr('classif.acc')

# run spsa
spsaMod <- spFeatureSelection( task = task,
                               wrapper = wrapper,
                               scoring = measure,
                               num.features.selected = 3,
                               n.jobs = 1,
                               iters.max = 2,
                               num.grad.avg = 1)


# obtain summary
summary(spsaMod)

# plot spsaMod
plot(spsaMod)                                  # simplest plot
plot(spsaMod, errorBar = TRUE)                 # plot with error bars
plot(spsaMod, errorBar = TRUE, se = TRUE)      # plot with error bars based on se
plot(spsaMod, errorBar = TRUE, annotateBest = TRUE)  # annotate best value
plot(spsaMod, errorBar = TRUE, ylab = 'Acc measure', type = 'o')

# obtain the wrapped model with the best performing features
bestMod <- getBestModel(spsaMod)

# predict using the best mod
pred <- bestMod$predict( task = spsaMod$task.spfs )

# Obtain confusion matrix
pred$confusion

# Get the importance ranks of best performing features
```

```
getImportance(spsaMod)
plotImportance(spsaMod)
```

---

spFSR.default *Default Function of SP-FSR for Feature Selection and Ranking*

Description

This is the default function of spFeatureSelection. See spFeatureSelection for example.

Usage

```
## Default S3 method:
spFSR(
  task,
  wrapper = NULL,
  scoring = NULL,
  perturb.amount = 0.05,
  gain.min = 0.01,
  gain.max = 2,
  change.min = 0,
  change.max = 0.2,
  bb.bottom.threshold = 10^(-8),
  mon.gain.A = 100,
  mon.gain.a = 0.75,
  mon.gain.alpha = 0.6,
  hot.start.num.ft.factor = 15,
  hot.start.max.auto.num.ft = 150,
  use.hot.start = TRUE,
  hot.start.range = 0.2,
  rf.n.estimators = 50,
  gain.type = "bb",
  num.features.selected = 0L,
  iters.max = 100L,
  stall.limit = 35L,
  n.samples.max = 5000,
  ft.weighting = FALSE,
  encoding.type = "encode",
  is.debug = FALSE,
  stall.tolerance = 10^(-8),
  random.state = 1,
  rounding = 3,
  run.parallel = TRUE,
```

```
  n.jobs = NULL,
  show.info = TRUE,
  print.freq = 10L,
  num.cv.folds = 5L,
  num.cv.reps.eval = 3L,
  num.cv.reps.grad = 1L,
  num.grad.avg = 4L,
  perf.eval.method = "cv"
)
```

Arguments

task	A task tsk object created using **mlr3** package. It must be either a ClassifTask or RegrTask object.
wrapper	A Learner lrn object created using **mlr3** package or a GraphLearner object created using **mlr3pipelines** package. Multiple learners object is not supported. If left empty will select random forest by default.
scoring	A performance measure msr within the **mlr3** package supported by the task. If left blank will select accuracy for classification and r-squared for regression.
perturb.amount	Perturbation amount for feature importances during gradient approximation. It must be a value between 0.01 and 0.1. Default value is 0.05.
gain.min	The minimum gain value. It must be greater than or equal to 0.001. Default value is 0.01.
gain.max	The maximum gain value. It must be greater than or equal to gain.min. Default value is 1.0.
change.min	The minimum change value. It must be non-negative. Default value is 0.0.
change.max	The maximum change value. It must be greater than change.min. Default is 0.2.
bb.bottom.threshold	The threshold value of denominator for the Barzilai-Borwein gain sequence. It must be positive. Default is 1/10^8.
mon.gain.A	Parameter for the monetone gain sequence. It must be a positive integer. Default is 100.
mon.gain.a	Parameter for the monetone gain sequence. It must be positive. Default is 0.75.
mon.gain.alpha	Parameter for the monetone gain sequence. It must be between (0, 1). Default is 0.6.
hot.start.num.ft.factor	The factor of features to select for hot start. Must be an integer greater than 1. Default is 15.
hot.start.max.auto.num.ft	The maximum initial number of features for automatic hot start. Must be an integer greater than 1. Default is 75.
use.hot.start	Logical argument. Whether hot start should be used. Default is True.
hot.start.range	Float, the initial range of imputations carried over from hot start. It must be between (0,1). Default is 0.2.

`rf.n.estimators`
: integer, The number of trees to use in the random forest hot start. The default is 50.

`gain.type`
: The gain sequence to use. Accepted methods are 'bb' for Barzilai-Borwein or 'mon' for a monetonic gain sequence. Default is 'bb'.

`num.features.selected`
: Number of features selected. It must be a nonnegative integer and must not exceed the total number of features in the task. A value of 0 results in automatic feature selection. Default value is 0L.

`iters.max`
: Maximum number of iterations to execute. The minimum value is 2L. Default value is 300L.

`stall.limit`
: Number of iterations to stall, that is, to continue without at least `stall.tolerance` improvement to the measure value. The mininum value is 2L. Default value is 100L.

`n.samples.max`
: The maximum number of samples to select from sampling. It must be a non-negative integer. Default is 2500.

`ft.weighting`
: Logical argument. Include simultaneous feature weighting and selection?. Default is FALSE.

`encoding.type`
: Encoding method for factor features for feature weighting, default is 'encoded'.

`is.debug`
: Logical argument. Print additional debug messages? Default value is FALSE.

`stall.tolerance`
: Value of stall tolerance. It must be strictly positive. Default value is 1/10^8.

`random.state`
: random state used. Default is 1.

`rounding`
: The number of digits to round results. It must be a positive integer. Default value is 3.

`run.parallel`
: Logical argument. Perform cross-validations in parallel? Default value is TRUE.

`n.jobs`
: Number of cores to use in case of a parallel run. It must be less than or equal to the total number of cores on the host machine. If set to `NULL` when `run.parallel` is `TRUE`, it is taken as one less of the total number of cores.

`show.info`
: If set to `TRUE`, iteration information is displayed at print frequency.

`print.freq`
: Iteration information printing frequency. It must be a positive integer. Default value is 10L.

`num.cv.folds`
: The number of cross-validation folds when 'cv' is selected as `perf.eval.method`. The minimum value is 3L. Default value is 5L.

`num.cv.reps.eval`
: The number of cross-validation repetitions for feature subset evaluation. It must be a positive integer. Default value is 3L.

`num.cv.reps.grad`
: The number of cross-validation repetitions for gradient averaging. It must be a positive integer. Default value is 1L.

`num.grad.avg`
: Number of gradients to average for gradient approximation. It must be a positive integer. Default value is 4L.

`perf.eval.method`
: Performance evaluation method. It must be either 'cv' for cross-validation or 'resub' for resubstitution. Default is 'cv'.

Value

spFSR returns an object of class "spFSR". An object of class "spFSR" consists of the following:

task.spfs	An **mlr3** package tsk object defined on the best performing features.
wrapper	An **mlr3** package lrn object or a **mlr3pipelines** package GraphLearner object as specified by the user.
scoring	An **mlr3** package msr as specified by the user.
param best.model	An **mlr3** package model object trained by the wrapper using task.spfs.
iter.results	A data.frame object containing detailed information on each iteration.
features	Names of the best performing features.
num.features	The number of best performing features.
importance	A vector of importance ranks of the best performing features.
total.iters	The total number of iterations executed.
best.iter	The iteration where the best performing feature subset was encountered.
best.value	The best measure value encountered during execution.
best.std	The standard deviation corresponding to the best measure value encountered.
run.time	Total run time in minutes.
results	Dataframe with boolean of selected features, names and measure
call	Call.

References

David V. Akman et al. (2022) k-best feature selection and ranking via stochastic approximation, *Expert Systems with Applications*, **Vol. 213**. See doi:10.1016/j.eswa.2022.118864

G.F.A Yeo and V. Aksakalli (2021) A stochastic approximation approach to simultaneous feature weighting and selection for nearest neighbour learners, *Expert Systems with Applications*, **Vol. 185**. See doi:10.1016/j.eswa.2021.115671

See Also

spFeatureSelection.

---

summary.spFSR *Summarising a spFSR object*

Description

Summarising a spFSR object

Usage

```
## S3 method for class 'spFSR'
summary(object, ...)
```

Arguments

`object`	A `spFSR` object
`...`	Additional arguments

Value

Summary of a spFSR object consisting of number of features selected, wrapper type, total number of iterations, the best performing features, and the descriptive statistics of the best iteration result (the iteration where the best performing features are found).

See Also

getImportance, spFSR.default, and spFeatureSelection.

Index